Bài 5

Ngôn ngữ C
Biến, Phép toán, Biểu thức, Lệnh

Nội dung

- Kiểu dữ liệu cơ bản (type)
- Bộ hiệu chỉnh kiểu dữ liệu (modifier)
- Biến (variable)
- Bộ định tính biến (qualifier)
- Định danh (identifier)
- Phạm vi khả kiến của định danh (scope)
- Hằng (literal)
- Phép toán (operator)
- Biểu thức (expression)
- Lệnh (statement)

Kiểu dữ liệu cơ bản

- Ngôn ngữ C hổ trợ 5 kiểu cơ bản:
  - char – kí tự, kích thước 1 byte
  - int – số nguyên có dấu, kích thước 2 hoặc 4 byte
  - float – số thực có dấu, kích thước 4 byte
  - double – số thực có dấu, kích thước 8 byte
  - void – kiểu rỗng
- Kích thước và phạm vi của các kiểu có thể khác nhau giữa các bộ xử lý và trình biên dịch.
- Ngôn ngữ C chỉ qui định về phạm vi tối thiểu của kiểu.

Bộ hiệu chỉnh kiểu dữ liệu

- **Modifiers** là nhóm các từ dùng để báo cho trình biên dịch biết các đặc tính bổ sung của một kiểu cơ bản.
- Ngôn ngữ C cung cấp 4 từ hiệu chỉnh cho các kiểu cơ bản:
  - signed: có dấu
  - unsigned: không dấu
  - short: kích thước lưu trữ nhỏ
  - long: kích thước lưu trữ lớn

Biến - Khai báo

- Lệnh khai báo biến
  <kiểu> <tên biến>;

```
char c;
int i, j, k;
long double balance, profit, loss;
```

- Biến có thể được khai báo ở ba nơi:
  - Bên trong hàm (*biến cục bộ*).
  - Tiêu đề của hàm (*tham số hình thức*).
  - Bên ngoài tất cả các hàm (*biến toàn cục*).
- Biến phải được khai báo trước khi sử dụng.

Biến cục bộ

- Là biến được khai báo bên trong một hàm.
- Chỉ được dùng bởi các lệnh trong khối mã nơi nó được khai báo.
- Chỉ tồn tại trong khi khối mã đang thực thi.
- Biến được khai báo ở khối bên trong sẽ ẩn biến cùng tên được khai báo ở khối bên ngoài.

```
void func1() {
    int x;
    x = 10;
}

void main() {
    int x = 10;
    if (x == 10) {
        /* Biến x này che biến
           x ở khối ngoài */
        int x = 99;
        printf("Inner x: %d", x);
    }
    printf("Outer x: %d", x);
}
```

Tham số hình thức

- Nếu một hàm có sử dụng các đối số, ta phải khai báo các biến để nhận giá trị của các đối số. Các biến này được gọi là các tham số hình thức của hàm.

```
/* Hàm này trả về bình phương của một số
   nguyên x là tham số hình thức */
int sqr(int x) {
    return x * x;
}
```

- Tham số hình thức hoạt động như biến cục bộ bên trong hàm.

Biến toàn cục

- Là biến được khai báo bên ngoài tất cả các hàm.
- Được dùng bởi bất kỳ lệnh nào của chương trình.
- Tồn tại suốt quá trình chương trình thực thi.

```
#include <stdio.h>

/* count là biến toàn cục */
int count;

void func1() {
    printf("count = %d\n",
           count);
}

void main() {
    count = 100;
    func1();
}
```

Biến - Khởi tạo

- Lệnh khởi tạo biến

  <kiểu> <tên biến> = <hằng>;

```
char ch = 'a';
int first = 0;
double balance = 123.23;
```

- Biến toàn cục nếu không được khởi tạo sẽ mặc định được gán giá trị 0.
- Biến cục bộ nếu không được khởi tạo sẽ nhận giá trị rác trước khi thực hiện phép gán đầu tiên với nó.

Bộ định tính biến

- **Qualifiers** là nhóm các từ dùng để báo cho trình biên dịch biết cách thức các biến có thể được truy cập hay thay đổi giá trị.
- Ngôn ngữ C cung cấp 2 từ định tính:
  - const – nội dung của biến không thể thay đổi. Biến với định tính const được xem như hằng được đặt tên.

```
const int SALES_TAX_RATE = 0.1;
```

  - volatile – ngăn chặn sự thay đổi nội dung của biến một cách không báo trước, thường được sử dụng trong lập trình nhúng, lập trình đa luồng.

Định danh

- Định danh (identifier) là cách gọi chung tên của các biến, các hàm và các thành phần do người lập trình định nghĩa.
  - Chỉ gồm chữ cái, chữ số và dấu gạch _.
  - Kí tự đầu tiên phải là chữ cái hoặc dấu _.
- Có phân biệt chữ hoa, chữ thường.
- Không được trùng với các từ dành riêng của ngôn ngữ C.

Phạm vi khả kiến của định danh (1)

- Phạm vi trong chương trình mà một định danh được biết đến và có thể được tham chiếu.
- Ngôn ngữ C qui định 4 phạm vi khả kiến của định danh:
  - Phạm vi tập tin (file scope).
  - Phạm vi khối mã (block scope).
  - Phạm vi nguyên mẫu hàm (function prototype scope).
  - Phạm vi hàm (function scope).

Phạm vi khả kiến của định danh (2)

- Phạm vi tập tin
  - Bắt đầu ở đầu tập tin và kết thúc ở cuối tập tin.
  - Đề cập đến định danh được khai báo bên ngoài tất cả các hàm.
  - Định danh có phạm vi tập tin được nhìn thấy trong toàn bộ tập tin.
  - Biến có phạm vi tập tin là toàn cục.
- Phạm vi khối mã
  - Bắt đầu ở dấu mở khối { và kết thúc ở dấu đóng khối }.
  - Tham số của hàm được xem có phạm vi khối của hàm.
  - Biến có phạm vi khối là cục bộ trong khối mà nó được khai báo.

Phạm vi khả kiến của định danh (3)

- Phạm vi nguyên mẫu hàm
  - Đề cập đến định danh được khai báo trong nguyên mẫu hàm và chỉ có thể nhìn thấy trong nguyên mẫu hàm.
- Phạm vi hàm
  - Bắt đầu ở dấu mở { và kết thúc ở dấu đóng } khối mã của hàm.
  - Đề cập đến các nhãn (label) được khai báo bên trong hàm.
    - Một nhãn được sử dụng làm đích cho câu lệnh ***goto***.

Hằng

- Hằng (literal) là các giá trị nhất định (theo nghĩa đen) mà chương trình không thể thay đổi và có thể thuộc kiểu cơ bản bất kỳ.

Hằng	Kiểu	
`'A' 'f'`	char	Hằng kí tự là một kí tự được đặt trong cặp dấu ' '.
`"Hello"`	con trỏ char	Hằng chuỗi kí tự là một dãy các kí tự được đặt trong cặp dấu " ".

`-123 21000`
`201000`
`1.23`
`3.14e-3`
	int
	long int
	double
	double
		Hằng số là các giá trị được biểu diễn ở dạng số nguyên hoặc số dấu phẩy động.

Hằng - Chỉ định kiểu

- Hằng nguyên, mặc định, trình biên dịch chọn một kiểu tương thích có phạm vi nhỏ nhất phù hợp với hằng.
- Hằng dấu phẩy động mặc định có kiểu là double.
- Có thể chỉ định kiểu cho hằng bằng cách thêm kí tự F (float), L (long), U (unsigned) vào sau hằng.

Hằng	Kiểu
`10000U`	unsigned int
`21000L`	long int
`1.23L`	long double
`1.23F 3.14e-3f`	float

Phép toán

- Ngôn ngữ C cung cấp 4 nhóm phép toán chính:
  - Phép toán số học.
  - Phép toán quan hệ (so sánh).
  - Phép toán logic (luận lý).
  - Phép toán bit.
- Ngoài ra còn có một số phép toán đặc biệt
  - Phép gán, phép ép kiểu, …

Phép gán

- Dùng để lưu giá trị vào biến.
  <biến> = <biểu thức>

```
char ch;
int x = 65, y;
float f = 62.5;
y = x + 200;   // Kết quả y là 265
ch = y;        // Kết quả ch là 9
x = f;         // Kết quả x là 62
f = y;         // Kết quả f là 265.0
```

- Có thể dùng nhiều phép gán cùng lúc

```
x = y = z = 0
```

Phép gán - Chuyển kiểu

- Trình biên dịch tự động chuyển kiểu giá trị của *<biểu thức>* sang kiểu của *<biến>*.
- Một số dạng chuyển kiểu phổ biến.

Kiểu biến	Kiểu biểu thức	Khả năng mất thông tin
signed char	char	Nếu biểu thức > 127, biến là âm
char	int	8 bit bên trái
char	long int	24 bit bên trái
int	long int	16 bit bên trái
int	float	phần thập phân
float	double	độ chính xác

Phép gán phức hợp

- Các phép gán với biểu thức có dạng
  var = var <phép toán hai ngôi> <biểu thức>
  có thể viết theo dạng phức hợp
  var <phép toán hai ngôi>= <biểu thức>

Dạng thường	Dạng phức hợp
`x = x + 5`	`x += 5`
`x = x - 5`	`x -= 5`
`x = x * 5`	`x *= 5`
`x = x / 5`	`x /= 5`

Phép toán số học

- Ngôn ngữ C có các phép toán số học sau:

Phép toán	Thao tác
-	Trừ hoặc trừ một ngôi (đổi dấu)
+	Cộng
*	Nhân
/	Chia (với kiểu float, double) hoặc chia nguyên (với kiểu int, char)
%	Lấy số dư trong phép chia nguyên
--	Giảm đơn vị
++	Tăng đơn vị

Phép toán số học - Minh họa

Phép toán	Minh họa
Trừ một ngôi	`x = -y`
Chia (với kiểu int, char)	`5 / 2` (kết quả là 2)
Chia lấy số dư (chỉ dùng với kiểu int, char)	`5 % 2` (kết quả là 1)
Giảm đơn vị	`--x` hoặc `x--` (tương đương `x = x - 1`)
Tăng đơn vị	`++x` hoặc `x++` (tương đương `x = x + 1`)

Phép toán số học - Độ ưu tiên

Độ ưu tiên (cao xuống thấp)	Tính liên kết
++ --	trái sang phải
- (trừ một ngôi)	trái sang phải
* / %	trái sang phải
+ -	trái sang phải

```
x = 10
y = -1 + 5 % 2 + ++x
    #2 (-1)  #3 (1)   #1 (11)
       #4 (0)
          #5 (11)
```


Phép toán quan hệ

- Ngôn ngữ C cung cấp các phép toán quan hệ sau

Phép toán	Thao tác
>	So sánh lớn hơn
>=	So sánh lớn hơn hoặc bằng
<	So sánh nhỏ hơn
<=	So sánh nhỏ hơn hoặc bằng
==	So sánh bằng
!=	So sánh không bằng

- Kết quả của một phép toán quan hệ là một trong hai giá trị **khác 0 (ĐÚNG)** và **bằng 0 (SAI)**.

Phép toán logic

- Ngôn ngữ C cung cấp các phép toán logic sau

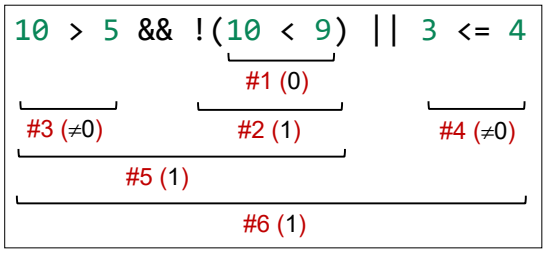
Phép toán	Thao tác
&&	AND logic
||	OR logic
!	NOT logic

- Kết quả của một phép toán logic là một trong hai giá trị **bằng 1 (ĐÚNG)** và **bằng 0 (SAI)**.

p	q	p && q	p || q	!p
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

Phép toán quan hệ, logic - Độ ưu tiên

Độ ưu tiên (cao xuống thấp)	Tính liên kết
!	trái sang phải
> >= < <=	trái sang phải
== !=	trái sang phải
&&	trái sang phải
||	trái sang phải



Phép toán bit (1)

- Ngôn ngữ C cung cấp các phép toán bit sau

Phép toán	Thao tác
&	AND bit
|	OR bit
^	XOR bit
~	Đảo bit
>>	Dịch phải bit
<<	Dịch trái bít

- Thực hiện thao tác kiểm tra, thiết lập hoặc dịch chuyển các bit thật của một vùng nhớ có kích thước 1 byte hoặc 1 word (2 bytes) ứng với các kiểu cơ bản char, int.

Phép toán bit (2)

- Kết quả của các phép toán bit là 1 hoặc 0.

p	q	p & q	p | q	~p	p ^ q
0	0	0	0	1	0
0	1	0	1	1	1
1	0	0	1	0	1
1	1	1	1	0	0

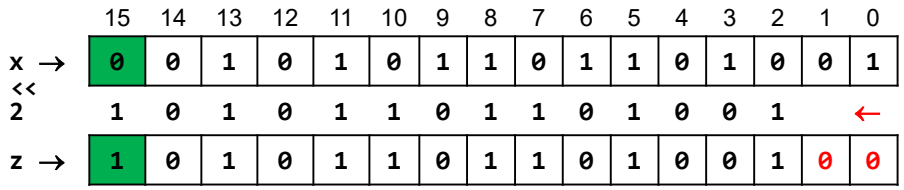
```
int x = 11113, y = 5550, z;
z = x & y;      // Kết quả z là 296
```

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
x →	0	0	1	0	1	0	1	1	0	1	1	0	1	0	0	1
& y →	0	0	0	1	0	1	0	1	1	0	1	0	1	1	1	0
z →	0	0	0	0	0	0	0	1	0	0	1	0	1	0	0	0

Phép toán bit (3)

- Phép toán <<

```
int x = 11113, z;
z = x << 2;      // Kết quả z là -21048
```

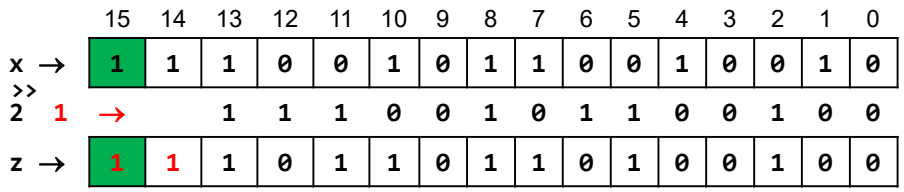


- n << k tương đương n * 2^k
  - Trường hợp trên z = -21048 do 11113 * 2^2 = 44452 vượt phạm vi kiểu int của biến z.

Phép toán bit (4)

- Phép toán >>

```
int x = -6766, z;
z = x >> 2;      // Kết quả z là -1692
```



- n >> k tương đương n / 2^k.

Một số phép toán khác

- Ngôn ngữ C còn cung cấp một số phép toán tiện dụng khác

Phép toán	Thao tác
`()`	Tăng độ ưu tiên cho các phép toán
`& *`	Thao tác trên các con trỏ
`[]`	Truy xuất phần tử mảng theo chỉ mục
`?:`	Thay thế cho một dạng lệnh lựa chọn
`,`	Xâu chuỗi các biểu thức `x = (y = 3, y + 1)` kết quả x là 4
`(type)`	Ép kiểu

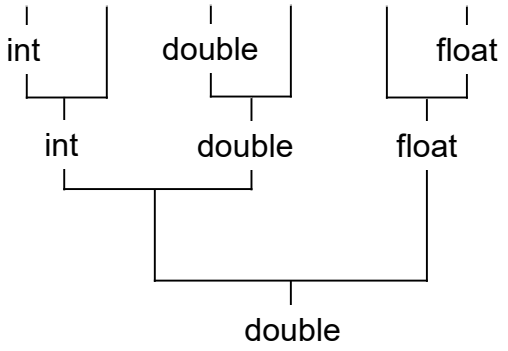
Biểu thức

- Biểu thức là sự kết hợp hợp lệ của các phép toán, các hằng, các biến và các hàm.
- Giá trị của biểu thức được ước tính dựa trên độ ưu tiên và tính liên kết của các phép toán.
- Chuyển kiểu biểu thức khi ước tính giá trị
  - Các toán hạng (biến, hằng, hàm) có kiểu khác nhau thì trình biên dịch chuyển chúng thành cùng một kiểu.
  - Qui tắc thăng cấp kiểu - toán hạng có kiểu thấp hơn sẽ được chuyển về cùng kiểu với toán hạng có kiểu cao hơn.
- Ép kiểu biểu thức bằng phép toán `(type)` theo dạng

  (kiểu) <biểu thức>

Biểu thức - Chuyển kiểu

```
char ch;
int i;
float f;
double d;
x = (ch / i) + (f * d) - (f + i);
```



Lệnh

- Lệnh biểu thức (expression statement) là một biểu thức hợp lệ theo sau là **dấu ;**
- Lệnh khối (block statement) là nhóm các lệnh bắt đầu bằng **dấu mở khối {** và kết thúc bằng **dấu đóng khối }** tương ứng.

```
getRectangle(); // lệnh gọi hàm
a = b + c;      // lệnh gán
a + b;          // hợp lệ nhưng vô nghĩa
;               // lệnh rỗng
{ }             // lệnh khối
```

- Lệnh điều khiển (control statement).

Tham khảo - Kiểu dữ liệu C chuẩn (1)

Kiểu	Cỡ kiểu theo bit	Phạm vi tối thiểu
char	8	-127 ... 123
unsigned char	8	0 ... 255
signed char	8	-127 ... 127
int	16 hoặc 32	-32.767 ... 32.767
unsigned int	16 hoặc 32	0 ... 65.535
signed int	16 hoặc 32	tương tự **int**
short int	16	-32.767 ... 32.767
unsigned short int	16	0 ... 65.535
signed short int	16	tương tự **short int**
long int	32	-2.147.483.647 ... 2.147.483.647

Tham khảo - Kiểu dữ liệu C chuẩn (2)

Kiểu	Cỡ kiểu theo bit	Phạm vi tối thiểu
long long int	64	$-(2^{63} - 1) ... 2^{63} - 1$
signed long int	32	tương tự **long int**
unsigned long int	32	0 ... 4.294.967.295
unsigned long long int	64	$2^{64} - 1$
float	32	1e - 37 ... 1e + 37 (6 chữ số thập phân)
double	64	1e - 37 ... 1e + 37 (10 chữ số thập phân)
long double	80	1e - 37 ... 1e + 37 (10 chữ số thập phân)

Tham khảo - Thứ tự phép toán (1)

Độ ưu tiên (cao xuống thấp)	Tính liên kết

`() [] -> .`
`! ~ ++ -- - (type) * &`
`* / %`
`+ -`
`<< >>`
`< <= > >=`
`== !=`
	- Tất cả các phép toán, ngoại trừ các phép toán một ngôi (- * &), các phép toán gán, phép toán ?:, liên kết từ trái sang phải.

Tham khảo - Thứ tự phép toán (2)

Độ ưu tiên (cao xuống thấp)	Tính liên kết
& (AND bit) ^ | && || ?: = += -= *= /= %= ... ,	- Các phép toán một ngôi (- * &), các phép toán gán, phép toán ?: liên kết từ phải sang trái.

Tham khảo - Bảng mã ASCII chuẩn

0	*NUL*	16	*DLE*	32	SPC	48	0	64	@	80	P	96	`	112	p
1	*SOH*	17	*DC1*	33	!	49	1	65	A	81	Q	97	a	113	q
2	*STX*	18	*DC2*	34	"	50	2	66	B	82	R	98	b	114	r
3	*ETX*	19	*DC3*	35	#	51	3	67	C	83	S	99	c	115	s
4	*EOT*	20	*DC4*	36	$	52	4	68	D	84	T	100	d	116	t
5	*ENQ*	21	*NAK*	37	%	53	5	69	E	85	U	101	e	117	u
6	*ACK*	22	*SYN*	38	&	54	6	70	F	86	V	102	f	118	v
7	*BEL*	23	*ETB*	39	'	55	7	71	G	87	W	103	g	119	w
8	*BS*	24	*CAN*	40	(	56	8	72	H	88	X	104	h	120	x
9	*HT*	25	*EM*	41	)	57	9	73	I	89	Y	105	i	121	y
10	*LF*	26	*SUB*	42	*	58	:	74	J	90	Z	106	j	122	z
11	*VT*	27	*ESC*	43	+	59	;	75	K	91	[	107	k	123	{
12	*FF*	28	*FS*	44	,	60	<	76	L	92	\	108	l	124	|
13	*CR*	29	*GS*	45	-	61	=	77	M	93	]	109	m	125	}
14	*SO*	30	*RS*	46	.	62	>	78	N	94	^	110	n	126	~
15	*SI*	31	*US*	47	/	63	?	79	O	95	_	111	o	127	DEL